CLAIMS

What is claimed is:

1. A method comprising:

disregarding a received load instruction when a currently active load operation is detected;

otherwise, directing, in response to the received load instruction, a memory protection element to form a secure memory environment including one or more protected memory regions, such that unauthorized read/write access to the one or more protected memory regions is prohibited; and

storing, within a digest information repository, a cryptographic hash value of the one or more protected memory regions, thereby enabling establishment of security verification of the secure memory environment.

2. The method of claim 1, wherein disregarding further comprises:

receiving a load secure region instruction as the load instruction from security initiation software, such that the load secure region instruction indicates the one or more memory regions to be established as the secure memory environment;

verifying that the security initiation software possesses a sufficient privilege level to issue the load secure region instruction;

once verified, determining whether a currently active load secure region operation exclusivity mutex exists;

when a currently active load secure region operation exclusivity mutex is detected, disregarding the received load secure region instruction request; and

otherwise establishing a load secure region instruction exclusivity mutex.

3. The method of claim 1, wherein disregarding further comprises:

determining whether a secure reset command has been received; and

when a secure reset command has been received, disregarding the received load instruction request.

4. The method of claim 3, wherein determining further comprises:

querying values within the digest information repository; and

when values within the digest information repository are committed, detecting receipt of a reset command.

5. The method of claim 1, wherein directing further comprises:

determining the one or more protected memory regions from the received load instruction request; and

providing a request, including parameters of the one or more protected memory regions, to a memory protection element, such that the memory protection element establishes the secure memory environment requested by a load secure region operation according to the load instruction.

6. The method of claim 1, wherein direct further comprises:

clearing the digest information repository; and

recording secure environment information within the digest information repository.

7. The method of claim 1, wherein storing further comprises:

hashing contents of the secure memory region to generate a cryptographic hash identification value as a secure software identification value for software contained within the secure environment; and

storing the secure software identification value within the digest information repository; and

releasing a load operation exclusivity mutex.

8. The method of claim 1, further comprising:

receiving a security authentication request from an outside agent;

accessing, via a secure channel, a secure software identification value;

digitally signing the software identification value; and

transmitting the digitally signed secure software identification value to the outside agent, such that the outside agent may examine a component state of secure software within the secure memory environment and decide whether to trust the secure software.

9. The method of claim 1, further comprising:

issuing a processor reset request to a system chipset; and

resetting, by the chipset, all processors coupled to the chipset in response to the processor reset request.

10. A method comprising:

resetting, in response to a secure reset request, one or more processors within a system, the secure reset request issued following completion of a load secure region operation; and

enabling processor read/write access to one or more protected memory regions of a secure memory environment formed in accordance with a load secure region instruction request to establish a secure environment.

11. The method of claim 10, wherein resetting further comprises:

issuing, to a system chipset platform, a processor reset command; and

issuing, by the chipset platform, a reset request to each of the one or more processors.

12. The method of claim 11, wherein issuing further comprises:

performing a data write to an input/output port address; and

decoding, by the chipset platform, the data write to detect a processor reset command.

13. The method of claim 10, wherein re-enabling further comprises:

committing data within a digest information repository to indicate a currently active reset command;

causing each of the processors coupled to the system to enter a known privilege state; and

directing a memory protection element to re-enable processor read/write access to the secure memory environment.

14. The method of claim 13, wherein causing comprises:

setting data storage elements within the one or more processors to one of the constant values stored in/generated by the processor and values derived from parameters specified by security initiation software at issuance of the load secure region instruction.

15. A computer readable storage medium including program instructions that direct a computer to function in a specified manner when executed by a processor, the program instructions comprising:

disregarding a received load instruction when a currently active load instruction is detected;

otherwise, directing, in response to the received load instruction, a memory protection element to form a secure memory environment including one or more protected memory regions, such that unauthorized read/write access to the one or more protected memory regions is prohibited; and

storing, within a digest information repository, a cryptographic hash value of the one or more protected memory regions, thereby enabling establishment of security verification of the secure environment.

16. The computer readable storage medium of claim 15, wherein disregarding further comprises:

receiving an instruction as the load secure region instruction from security initiation software, such that the load secure region instruction indicates the one or more memory regions to be established as the secure memory environment, as well as secure software loaded within the secure memory environment;

verifying that the security initiation software possesses a sufficient privilege level to issue the load secure region instruction;

once verified, determining whether a currently active load operation exclusivity mutex exists;

when a currently active load operation exclusivity mutex is detected, disregarding the received load secure region instruction request; and

otherwise establishing a load secure region operation exclusivity mutex.

17. The computer readable storage medium of claim 15, wherein disregarding further comprises:

determining whether a secure reset command has been received; and

when a secure reset command has been received, disregarding the received load instruction request.

18. The computer readable storage medium of claim 17, wherein determining further comprises:

querying values within a digest information repository; and

when values within the digest information repository are committed, detecting receipt of a reset command.

19. The computer readable storage medium of claim 15, wherein directing further comprises:

determining the one or more protected memory regions from the received load instruction request; and providing a request including parameters of the one or more protected memory regions to a memory protection element, such that the memory protection element establishes the secure memory region requested by a load secure region operation according to the load instruction.

20. The computer readable storage medium of claim 15, wherein programming further comprises:

clearing the digest information repository; and

recording security environment information within the digest information repository.

21. The computer readable storage medium of claim 15, wherein storing further comprises:

hashing contents within the secure memory environment to generate a cryptographic hash value as a secure software identification value for software loaded within the secure memory environment;

storing the secure software identification value within the digest information repository; and

releasing a load operation exclusivity mutex.

22. The computer readable storage medium of claim 15, further comprising:

receiving a security authentication request from an outside agent;

accessing, via a secure channel, a software identification value;

digitally signing the software hash identification value; and

transmitting the digitally signed software identification value to the outside agent, such that the outside agent may examine a component state of secure software within the secure memory environment and decide whether to trust the secure software.

23. The computer readable storage medium of claim 15, further comprising:

issuing a processor reset request to a system chipset; and

resetting, by the chipset, all processors coupled to the chipset in response to the processor reset request.

24. A computer readable storage medium comprising:

resetting, in response to a secure reset request, one or more processors within a system, the secure reset request issued following completion of a load secure region operation; and re-enabling processor read/write access to one or more protected memory regions of a secure memory environment formed in accordance with a load secure region instruction to establish a secure environment.

25. The computer readable storage medium of claim 24, wherein resetting further comprises:

issuing, to a system chipset platform, a processor reset command; and

issuing, by the chipset platform, a reset request to each of the one or more processors.

26. The computer readable storage medium of claim 25, wherein issuing further comprises:

performing a data write to an input/output port address; and

decoding, by the chipset platform, the data right to detect a processor reset command.

27. The computer readable storage medium of claim 24, wherein re-enabling further comprises:

committing data within a digest information repository to indicate a currently active reset command;

causing each of the processors coupled to the system to enter a known privilege state; and

programming a memory protection element to re-enable processor read/write to the secure memory environment.

28. The computer readable storage medium of claim 27, wherein each of the processors further comprises:

setting data storage elements within the processor to one of the constant values stored in/generated by the processor and values derived from parameters specified by security initiation software at issuance of the load secure region instruction.

29. An apparatus, comprising:

a memory controller;

a plurality of processors, each coupled to the memory controller and each having circuitry to execute instructions;

a memory protection element coupled to the memory controller, the memory protection element to block unauthorized memory access to one or more protected memory regions;

a digest information repository coupled to the memory controller to store a secure software identification value; and

a memory device coupled to the memory controller, having sequences of instructions stored therein, including at least one loading instruction, which when executed by a processor cause the processor to:

disregard a received load secure region instruction when a currently active load secure region operation is detected,

otherwise, direct, in accordance with a load secure region operation corresponding to the received load secure region instruction, a memory protection element to form a secure memory environment including one or more protected memory regions, such that unauthorized read/write access to the one or more protected memory regions is prohibited, and

store, according to the load secure region operation, within the digest information repository, a cryptographic hash value of the one or more protected memory regions as the secure software identification value, thereby enabling establishment of security verification of the secure memory environment by an outside agent.

30. The apparatus of claim 29, wherein the processor is further caused to:

receive a security authentication request from an outside agent;

access, via a secure channel, a secure software identification value;

digitally sign the secure software identification value; and

transmit the digitally signed secure software identification value to the outside agent, such that the outside agent may examine a component state of secure software within the secure memory environment and decide whether to trust the secure software.

31. The apparatus of claim 29, wherein the processor is further caused to:

reset, in response to a secure reset request, each of the plurality of processors, the secure reset request issued following completion of the load instruction; and

re-enable processor read/write access to the one or more protected memory regions of the secure memory environment formed in accordance with the load secure region operation to establish a secure environment.

32. The apparatus of claim 29, wherein the instruction to re-enable further causes the processor to:

commit data within a digest information repository to indicate a currently active secure reset command;

cause each of the processors coupled to the system to enter a known privilege state; and

direct a memory protection element to re-enable processor read/write access to the secure memory environment.

33. The apparatus of claim 32, wherein the instruction to cause processors to enter a privilege state further causes the processor to:

set data storage elements within the plurality of processors to one of the constant values stored in/generated by the processor and values derived from parameters specified by security initiation software at issuance of the load secure region instruction.

34. A system comprises:

a memory controller;

a plurality of processors, each coupled to the memory controller and each having circuitry to execute instructions;

a memory protection element coupled to the memory controller, the memory protection element to block unauthorized memory access to one or more protected memory regions;

an I/O controller coupled to the memory controller;

a verification unit coupled to the I/O controller and including a digest information repository to store a secure software identification value and to provide security verification to an outside agent; and

a storage device coupled to the memory controller, having sequences of instructions stored therein including at least one load instruction, which when executed by a processor causes the processor to:

disregard a received load secure region operation when a currently active load instruction is detected,

otherwise, direct, in accordance with a load secure region operation corresponding to the received load instruction, a memory protection element to form a secure memory environment including one or more protected memory regions, such that unauthorized read/write access to the one or more protected memory regions is prohibited, and

store, within the digest information repository, a cryptographic hash value of the one or more protected memory regions, thereby enabling establishment of security verification of the secure memory environment by an outside agent.

35. The system of claim 34, wherein the processor is further caused to:

receive a security authentication request from an outside agent;

access, via a secure channel, a secure software identification value;

digitally sign the secure software identification value; and

transmit the digitally signed secure software identification value to the outside agent, such that the outside agent may examine a component state of secure software within the secure memory environment and decide whether to trust the secure software.

36. The system of claim 34, wherein the processors is further caused to:

reset, in response to a secure reset request, each of the plurality of processors, the secure reset request issued following completion of the load secure region instruction request; and

re-enable processor read/write access to the one or more protected memory regions of the secure memory environment formed in accordance with the load secure region operation to establish a secure environment.

37. The system of claim 34, wherein the instruction to re-enable further causes the processor to:

commit data within a digest information repository to indicate a currently active secure reset command;

cause each of the plurality of processors to enter a known privilege state; and

direct a memory protection element to re-enable processor read/write access to the secure memory environment.

38. The system of claim 37, wherein the instruction to cause processors to enter a privilege state further causes the processor to:

set data storage elements within the plurality of processors to one of the constant values stored in/generated by the processor and values derived from parameters specified by security initiation software at issuance of the load secure region instruction.